100202520-1

UNITED STATES PATENT APPLICATION FOR

Method and Apparatus for Computing, Storing, And Multiplexing Modified Bytes In A Packet Router

Inventors:

Bruce E. LaVigne

Lewis S. Kootstra

Method and Apparatus for Computing, Storing, And Multiplexing Modified Bytes In A Packet Router

Technical Field

The present invention pertains to an apparatus and method for computing, storing, and multiplexing modified bytes in a packet router.

Background Art

Businesses and individuals rely upon the Internet and various communications networks for exchanging data electronically. Computers, personal digital assistants, and other types of mobile communication devices coupled to the Internet, enable users to readily gain access to and exchange data of all types (e.g., sound, text, numerical data, video, graphics, multi-media, emails, etc.) with other computers, databases, websites, etc. Now, users can participate in live discussions in chat rooms, play games in real-time, watch streaming video, listen to music, shop and trade on-line, download files, etc. And given the requisite bandwidth, it is possible to provide video-on-demand, HDTV, IP telephony, video teleconferencing, and other types of bandwidth intensive applications.

At the core of networking technology lies the fundamental proposition of digitizing the data into a string of bits comprised of one's and zero's. The digital data is divided into groups known as "packets." Embedded within each packet are various addresses mandated by an Internet Protocol (IP). These IP addresses are analogous to a postal letter's return address and recipient address in that the IP addresses provide information on how the packets are to be forwarded. Routers coupled to

various nodes of the network, examine the packets and route the packets based on their IP addresses. During transmission, a packet can be routed via multiple intermediary routers before reaching its final destination. In this manner, packets are separately transmitted and routed through the network until they eventually reach their intended destination.

Each time a router examines a packet and subsequently routes the packet to a different location, it must first essentially modify that packet. Specifically, the router must modify the Destination Address (DA), the Source Address (SA), the Time-to-Live (TTL), and the Checksum. One way by which this is commonly done today entails the use of software. A computer program takes the packet, modifies it, and then sends out the modified packet. Although this software approach is straightforward, it is relatively time intensive. Given that routers are being designed to route millions of packets per second, even the slightest incremental delay incurred while routing a packet can result in severe aggregate performance degradation.

In an effort to improve performance, some router manufacturers have implemented hardware designs. Rather than running software on a generic processor, these manufacturers have customized circuitry specifically designed to handle the requisite packet modifications. The specialized hardware reads the old values, computes the new values, writes the new values back to the packet memory, and then sends the modified packet out. One disadvantage to this hardware approach is that the additional circuitry increases complexity and resultant cost. Additional die space is necessary to fabricate the hardware design. A bigger

chip directly translates into increased production costs. Furthermore, re-writing the packet memory prior to the packet being sent, requires the addition of more write ports or packet buffers to temporarily hold the packet data. This adds to the complexity and cost. And although the packet modifications can be computed on-the-fly by hardware, the associated latency can detrimentally impact its performance. Ofcourse, this latency can be effectively masked through the use of pipelining techniques. However, the trade-off is that pipelining entails additional logic which, in turn, leads to increased production cost. This is especially the case with using pipelining to modify packets; a deep pipeline is needed in order to compute the Checksum because the Checksum must cover the entire portion of the packet being modified. Deep pipelines consume valuable chip space and can significantly increase the cost of the router.

Therefore, there are various disadvantages inherent to the prior art networking methods used to modify packets in routing.

SUMMARY OF THE INVENTION

A method and apparatus are disclosed for generating a modified packet for output from a router. First, a received packet is stored in one memory location. Modified bytes corresponding to the received packet are computed and stored in a separate memory location. The modified packet is generated by multiplexing between select unmodified bytes of the received packet with the modified bytes.

BRIEF DESCRIPTION OF THE DRAWINGS

The accompanying drawings, which are incorporated in and form a part of this specification, illustrate embodiments of the invention and, together with the description, serve to explain the principles of the invention:

Figure 1 shows an exemplary packet and its associated fields.

Figure 2 shows an exemplary circuit which can be used to store, compute, and multiplex modified bytes of information for routing in accordance with one embodiment of the present invention.

Figure 3 is a flowchart describing the process of computing, multiplexing, and outputting modified packets in accordance with one embodiment of the present invention.

Figure 4 shows a flowchart which describes the steps for selectively outputting bytes of a packet in a router according to one embodiment of the present invention.

Figure 5 shows a flow diagram describing in detail how received packets are modified and output according to one embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

The present invention relates to an apparatus and method for computing, storing, and multiplexing modified bytes in a packet router. In one embodiment, when a packet is received, the entire packet is stored in a packet buffer. The router identifies the locations of the fields within the packet which need to be modified. For each of these fields, new modified bytes of information are computed. The modified bytes are stored in a separate, dedicated memory. When a packet is sent out of the router, a multiplexer selectively outputs either the original bytes from the packet buffer or the modified bytes from the dedicated memory. The multiplexer makes its selection based on the location of the fields to be output. For instance, if the location of the field indicates that it need not be modified, the multiplexer selects the original bytes corresponding to that particular field contained in the packet buffer for output. But on the other hand, if the location of the field indicates that it needed to be modified, the multiplexer selects the modified bytes corresponding to that field contained in the dedicated memory for output. Thereby, the present invention enables packets to be output on-the-fly at extremely high data rates without the need for complicated, expensive pipelining stages. Furthermore, the present invention does not require additional write ports or extra packet buffers.

An example is now offered to describe one embodiment of the present invention in detail. Referring to Figure 1, an exemplary packet and its associated fields is shown. It can be seen that packet 101 contains a number of different fields. In particular, packet 101 includes a Destination Address field 102, a Source Address field 103, a Length field 104, an IP

Source Address 105, an IP Destination Address 106, a Time-to-Live field 107, a Checksum field 108, and a User Data field 109. The locations of each of these fields are specified according to a TCP/IP protocol. As such, the specific locations of each of the fields are pre-determined and known. In addition, the TCP/IP protocol specifies that several of the fields must be modified before the packet is allowed to be output by the router. In particular, the bytes of information contained in the Destination Address field 102, the Source Address field 103, the Time-to-Live field 107, and the Checksum field 108 must be modified in order for the packet to be properly routed. The Destination Address field 102 contains information that specifies the address of the next node to which the packet is being sent. This value is changed as the packet is forwarded from one node to the next. The Source Address field 103 contains information that specifies the address of the node that is sending the packet. Again, this value is updated as the packet is routed through various nodes of the network. The Time-to-Live (TTL) field 107 is designed to prevent packets from running in loops. Every router that handles a packet subtracts one from the packet's TTL. If the TTL reaches zero, the packet has expired and is discarded. As such, the value in the TTL field 107 must be modified by each router. The Checksum field 108 is used for error correction. It contains a checksum value which is representative of a running total based on the byte values transmitted in the packet. The receiver compares checksums to determine whether a transmission was error-free. And because other fields in the packet need be modified, the checksum value must correspondingly be modified as well.

Other fields, such as the Length 104, IP Source Address 105, the IP Destination Address 106, and User Data 109 fields also have fixed, pre-determined locations which are known. However, the information in these fields remain the same throughout the entire transmission process. For instance, the IP Source Address field 105 contains an address which uniquely identifies the sending device. This remains constant. Likewise, the address contained in the IP Destination Address field 106 remains constant because it uniquely identifies the intended receiving device. The Length field 104 contains a value which specifies the length of the packet. Some packets may have more user data, whereas other packets may have less user data. In any case, the length of the packet does not change throughout the transmission. And it is important to maintain the same user data contained in the User Data field 109.

As described above, the entire packet is initially stored in a packet buffer. In other words, the information contained in fields 102-109 are all stored in the packet buffer. However, the router computes a new value for each of the Destination Address 102, Source Address 103, Time-to-Live 107, and Checksum 108 fields. These new values can be several bytes wide. For example, the Destination Address and Source Address can be four, six, or more bytes wide. All of these newly computed values are stored in a dedicated memory separate from that of the packet buffer. When the packet is ready to be sent, a multiplexer selects between the packet buffer and the dedicated memory. Basically, the multiplexer is a switch that switches between original bytes stored in the packet buffer and the newly computed bytes stored in the dedicated memory. The selection is based on the particular field.

For instance, the packet header starts with the Destination Address field 102. Consequently, the multiplexer initially selects the bytes from the dedicated memory for forwarding. These bytes correspond to the newly computed destination address. It should be noted that the multiplexer does not select the bytes of the destination address stored in the packet buffer. Thereby, the old destination address is properly discarded. The next field is the Source Address field 103. Again, the multiplexer selects the dedicated memory from which to forward bytes of information. These bytes represent the newly computed source address. And the original source address from the packet buffer is discarded. When the Length field 104 is encountered, the multiplexer switches and selects the information contained in the packet buffer corresponding to the length of the packet. And due to the fact that the length of the packet is not modified, there is no length information stored in the dedicated memory. Likewise, the multiplexer forwards those bytes in the packet buffer corresponding to the IP Source Address field 105 and IP Destination field 106. The multiplexer then switches back to forwarding bytes of information from the dedicated memory when it reaches the TTL and Checksum fields 107 and 108. Lastly, the multiplexer switches again to forwarding bytes from the packet buffer for the duration of the User Data field(s) 109.

Figure 2 shows an exemplary circuit which can be used to store, compute, and multiplex modified bytes of information for routing in accordance with one embodiment of the present invention. The incoming packets are all stored in the packet buffer 201. Packet buffer 201 can be random access memory (RAM) including, but not limited to,

dynamic random access memory (DRAM) and/or static random access memory (SRAM). Other commonly used memories for performing the same function include first-in-first-out (FIFO) memory or input memory. A processor 202 calculates new values for those fields in the packets which need to be modified. In one embodiment, software can be used to calculate the modified bytes. In an alternative embodiment, the processor 202 can be specialized circuitry designed specifically to calculate the modified bytes. Furthermore, the modified bytes can be computed on-the-fly in real-time. Alternatively, the modified bytes can be pre-computed. As a first packet is in the process of being sent out, one or more subsequent packets can have their bytes modified. In other words, packets which are further down in the queue of the packet buffer 201 can be pre-computed. The pre-computed modified bytes are stored in memory 203 and are ready to be sent out at any time. Memory 203 can be DRAM, SRAM, etc. Memory 203 can store a number of modified bytes corresponding to different fields and also different packets. It should be noted that memory 203 is logically separate from that of packet buffer 201. The original packet data is not overwritten. Instead, the modified bytes are stored in a different memory area.

A multiplexer 204 selects bytes of information from either the packet buffer 201 or memory 203. For those fields which do not require any modification, the multiplexer 204 is directed by controller 205 to select the output from packet buffer 201. Otherwise, for those fields which do require modification, the multiplexer 204 is directed by controller 205 to select the output from memory 203. The re-assembled packet containing

the original bytes and modified bytes are then sent out of the multiplexer 204.

In one embodiment, controller 205 has information which specifies the locations of bytes of information which need to be modified. This is accomplished by supplying the particular format corresponding to the incoming packets (e.g., the fields, lengths, and/or locations). For example, the relative positions of the fields may change; the locations of the fields may be different; and the number of bytes corresponding to one or more fields can be varied. This formatting information enables the controller to adapt this embodiment of the present invention so that it can handle any type of incoming packets corresponding to present, past, or future protocols. It should be noted that multiplexer 204 can be done in software or can be replaced with a switch which selectively switches between the output from the packet buffer 201 and memory 203.

Figure 3 is a flowchart describing the process of computing, multiplexing, and outputting modified packets in accordance with one embodiment of the present invention. In step 301, incoming packets are buffered or otherwise temporarily stored. Certain bytes corresponding to the packet are then modified in step 302. In one embodiment, the modified bytes are pre-computed. The modified bytes are stored in a separate memory location in step 303. Note that the original packet is still retained in its unchanged state. The modified bytes are multiplexed together with select bytes from the original packet in step 304. The multiplexed output constitutes the modified packet which is ready for transmission by the router in step 305.

Figure 4 shows a flowchart which describes the steps for selectively outputting bytes of a packet in a router according to one embodiment of the present invention. Initially, incoming packets are received by the router in step 401. These packets are stored in a packet buffer in step 402. The fields of a packet which need to be modified in order to properly route that particular packet are identified in step 403. New bytes of information are then computed in step 404 corresponding to those fields which need to be modified. The modified bytes are stored in a separate memory area in step 405. When a packet is ready to be forwarded, a determination is made as to whether a particular field needed to be modified in step 406. If that field needed to be modified, the bytes corresponding to that particular field are selected from the separate memory area for output as described in step 407. Otherwise, step 408 shows that if the field did not need to be modified, the bytes of the original received packet corresponding to that particular field are selected for output. Step 409 results in all fields to be processed according to steps 406-408. In the end, the modified packet is transmitted, step 410.

Figure 5 shows a flow diagram describing in detail how received packets are modified and output according to one embodiment of the present invention. Packets are buffered in the order received, as depicted by 501. The bytes corresponding to certain fields of the received packets need to be modified. Only the modified bytes are stored in a separate memory location, as indicated by 502. In one embodiment, the present invention selects between certain bytes from the received packets 501 and others from the modified bytes 502 to formulate the modified output

packets. The modified output packets are represented in this figure by 503. In relation to the detailed descriptions given above, the received packets reside in the packet buffer; the modified bytes 502 are stored in a separate memory location; and the output packets 503 are output from the multiplexer.

More specifically, a first received packet (e.g., Packet0) contains a Destination Address (DA0), a Source Address (SA0), a Length (L0), an IP Source Address (IP SA0), an IP Destination Address (IP DA0), a Time-to-Live (TTL0), a Checksum (CKSM0), and User Data (Data0). A processor, dedicated hardware circuitry, and/or software is used to compute new values for those fields which need to be modified. Consequently for Packet0, the following modified bytes (mB0) are computed: a modified Destination Address (mDA0), a modified Source Address (mSA0), a modified Time-to-Live (mTTL0), and a modified Checksum (mCKSM0). The modified output packet (mPacket0) is formed by the controlled selection of values from either the received packets (Packet0) or modified bytes (mB0), depending upon whether that field needed to be modified.

Applying this rule, it can be seen that the Destination Address needed to be modified. As such, the first field of mPacket0 is selected from mB0. Consequently, the first field of mPacket0 is mDA0. Likewise, the second field needed to be modified. It follows then that the second field of mPacket0 is mSA0. However, the third field, did not need to be modified. Consequently, the third field of mPacket0 is selected from the received packets 501. The third field correctly contains L0. In similar fashion, the fourth and fifth fields of mPacket0 did not need to be modified. As a

result, the IP SA0 and IP DA0 are selected for output as part of mPacket0. But the mTTL0 and mCKSM are selected because they needed to be modified. And the integrity of the User Data needs to be maintained. Hence, User Data0 is appended to mPacket0. In summary, the modified output packet corresponding to the first received packet correctly contains a modified Destination Address (mDA0); a modified Source Address (mSA0); an unmodified Length (L0); an unmodified IP Source Address (IP SA0); an unmodified IP Destination Address (IP DA0); a modified TTL (mTTL0); a modified Checksum (mCKSM); and unmodified User Data (Data0).

This process can be repeated for successive packets. For example, modified bytes (mB1) can be pre-computed for a second packet (Packet1) to generate a modified output packet (mPacket1).

Thus, an apparatus and method for computing, storing, and multiplexing modified bytes in a packet router is described. While the present invention has been described in particular embodiments, it should be appreciated that the present invention should not be construed as limited by such embodiments, but rather construed according to the below claims.